

Programação em Python

Tópicos

- Iteração: instruções `while`, `for`, `break`

Exercícios

1. Para cada excerto de código abaixo, tente prever quantas iterações vão ser executadas e quais valores a ser impressos. Depois visualize a execução de cada excerto usando o [PythonTutor.com](https://www.python-tutor.com/).

<pre>n = 10 while n > 0: print(n) n -= 1</pre>	<pre>n = 1 while n < 1000: print(n) n *= 2</pre>
<pre>for n in (1, 2, 5, 10, 20, 50): print(n)</pre>	<pre>for c in "abcdefghijkl": print(c)</pre>
<pre>for n in range(20): print(n)</pre>	<pre>for n in range(50, 0, -5): print(n)</pre>

2. Escreva uma função `factorial(n)` que calcule o fatorial de n , definido por $n! = 1 \times 2 \times 3 \times \dots \times n$. Teste a função com alguns exemplos:

```
print(factorial(6))
print(factorial(4))
```

3. Pretende-se calcular a média de uma sequência de valores. Escreva um programa que peça ao utilizador uma sequência de números reais. Para terminar a sequência, o utilizador pressiona ENTER, introduzindo uma linha vazia. Nessa altura, o programa deve mostrar a média dos números introduzidos.
4. O seguinte programa mostra uma tabela dos quadrados de quatro números naturais. Implemente-o e experimente.

```
5. #Exercicio 4
6. # Quadrados dos primeiros 4 números
```

```
7. print("{:2s} {:2s}".format("n", "n2"))  
8. for n in [1, 2, 3, 4]:  
9.     print("{:2d} {:2d}".format(n, n**2))
```

- a) Modifique o programa para mostrar a tabela para números entre 1 e 20. Use a função `range`.
 - b) Acrescente uma coluna para mostrar 2^n . Use a mesma formatação que no exemplo dado.
5. ** Lembra-se da sequência de Fibonacci?

A sequência de Fibonacci é uma sequência de inteiros na qual cada elemento é igual à soma dos dois anteriores: 0, 1, 1, 2, 3, 5, 8, 13, ..., ou seja, cada termo obtém-se como $F_n = F_{n-1} + F_{n-2}$. Os primeiros valores são definidos como $F_0 = 0$ e $F_1 = 1$.

- a) Escreva uma função `Fibonacci(n)` para calcular o n-ésimo número de Fibonacci. *Sugestão: em cada iteração atualize e guarde os dois últimos valores da sequência.*
6. ** Escreva uma função `isPrime(n)` que devolva `True` se o número `n` é primo e `False`, caso contrário. Um número `N` é primo se não tiver divisores além de 1 e de `N`.

Relembre a abordagem seguida em aulas anteriores: tente dividir o número por 2, por 3, etc. Se encontrar um divisor exato, então o número não é primo.

Teste a função fazendo um programa que percorre todos os números entre 1 e 100 e indique para cada um se é primo ou não.